An introduction to Krivine realizability

Alexandre Miquel

UNIVERSIDAD DE LA REPUBLICA URUGUAY
FACULTAD DE INGENIERIA
EQUIPO DE LÓGICA UDELAR

April 28th, IMERL

What is classical realizability?

- Complete reformulation of the principles of Kleene realizability to take into account classical reasoning [Krivine 2009]
  - Based on Griffin's discovery about the connection between classical reasoning an control operators (call/cc)

$$\text{call/cc} \ : \ ((A \Rightarrow B) \Rightarrow A) \Rightarrow A \qquad \text{(Peirce's law)}$$

  - Interprets the Axiom of Dependent Choices (DC) [K. 2003]
- Initially designed for PA2, but extends to:
  - Higher-order arithmetic (PAω)
  - Zermelo-Fraenkel set theory (ZF) [K. 2001, 2012]
  - The calculus of inductive constructions (CIC) [M. 2007] (with classical logic in Prop)
- Deep connections with Cohen forcing [K. 2011]

  $\rightsquigarrow$ can be used to define new models of PA2/ZF [K. 2012]

Plan

Plan

The language of (minimal) second-order logic

- Second-order logic deals with two kinds of objects:
  - 1st-order objects = individuals (i.e. basic objects of the theory)
  - 2nd-order objects = k-ary relations over individuals

First-order terms and formulas

First-order terms	e, e'	$::=$	$x \mid f(e_1, \ldots, e_k)$
Formulas	A, B	$::=$	$X(e_1, \ldots, e_k) \mid A \Rightarrow B \mid \forall x\, A \mid \forall X\, A$

- Two kinds of variables
  - 1st-order vars: $x, y, z, \ldots$
  - 2nd-order vars: $X, Y, Z, \ldots$ of all arities $k \geq 0$
- Two kinds of substitution:
  - 1st-order subst.: $e\{x := e_0\}$, $A\{x := e_0\}$ (defined as usual)
  - 2nd-order subst.: $A\{X := P_0\}$, $P\{X := P_0\}$ (postponed)

First-order terms

- Defined from a first-order signature Σ (as usual):

First-order terms $\qquad e, e' \quad ::= \quad x \quad | \quad f(e_1, \ldots, e_k)$

  - f ranges over k-ary function symbols in Σ

- In what follows we assume that:

  1. Each k-ary function symbol f is interpreted in $\mathbb{N}$ by a function
  $$f^{\mathbb{N}} \;:\; \mathbb{N}^k \to \mathbb{N}$$
  2. The signature Σ contains at least a function symbol for every primitive recursive function (0, s, pred, $+$, $-$, $\times$, $/$, mod, $\ldots$), each of them being interpreted the standard way

- Denotation (in $\mathbb{N}$) of a closed first-order term e written $e^{\mathbb{N}}$

Formulas

- Formulas of minimal second-order logic

Formulas
$$A, B \quad ::= \quad X(e_1, \ldots, e_k) \quad | \quad A \Rightarrow B \quad | \quad \forall x\, A \quad | \quad \forall X\, A$$

only based on implication and 1st/2nd-order universal quantification

- Other connectives/quantifiers defined via second-order encodings:

$$\begin{array}{rcll}
\bot & \equiv & \forall Z\, Z & \text{(absurdity)} \\
\neg A & \equiv & A \Rightarrow \bot & \text{(negation)} \\
A \wedge B & \equiv & \forall Z\, ((A \Rightarrow B \Rightarrow Z) \Rightarrow Z) & \text{(conjunction)} \\
A \vee B & \equiv & \forall Z\, ((A \Rightarrow Z) \Rightarrow (B \Rightarrow Z) \Rightarrow Z) & \text{(disjunction)} \\
\exists x\, A(x) & \equiv & \forall Z\, (\forall x\, (A(x) \Rightarrow Z) \Rightarrow Z) & \text{(1st-order } \exists) \\
\exists X\, A(X) & \equiv & \forall Z\, (\forall X\, (A(X) \Rightarrow Z) \Rightarrow Z) & \text{(2nd-order } \exists) \\
e_1 = e_2 & \equiv & \forall Z\, (Z(e_1) \Rightarrow Z(e_2)) & \text{(Leibniz equality)}
\end{array}$$

Predicates

- Concrete relations are represented using predicates (syntactic sugar)

Predicates $P, Q \quad ::= \quad \hat{x}_1 \cdots \hat{x}_k A_0$ (of arity k)

Definition (Predicate application and 2nd-order substitution)

1. $P(e_1, \ldots, e_k)$ is the formula defined by

$$P(e_1, \ldots, e_k) \;\equiv\; A_0\{x_1 := e_1, \ldots, x_k := e_k\}$$

where $P \equiv \hat{x}_1 \cdots \hat{x}_k A_0$, and where $e_1, \ldots, e_k$ are k first-order terms

2. 2nd-order substitution $A\{X := P\}$ (where X and P are of the same arity k) consists to replace in the formula A every atomic sub-formula of the form

$$X(e_1, \ldots, e_k) \qquad \text{by the formula} \qquad P(e_1, \ldots, e_k)$$

- **Note:** Every k-ary 2nd-order variable X can be seen as a predicate:

$$X \;\equiv\; \hat{x}_1 \cdots \hat{x}_k X(x_1, \ldots, x_k)$$

Unary predicates as sets

- Unary predicates represent sets of individuals

  Syntactic sugar: $\{x : A\} \equiv \hat{x}A, \qquad e \in P \equiv P(e)$

Example: The set $\mathbb{N}$ of Dedekind numerals

$$\mathbb{N} \equiv \{x : \forall Z\,(0 \in Z \Rightarrow \forall y\,(y \in Z \Rightarrow s(y) \in Z) \Rightarrow x \in Z\}$$

- Relativized quantifications:

$$\begin{aligned}(\forall x \in P)\,A(x) &\equiv \forall x\,(x \in P \Rightarrow A(x))\\ (\exists x \in P)\,A(x) &\equiv \forall Z\,(\forall x\,(x \in P \Rightarrow A(x) \Rightarrow Z) \Rightarrow Z)\\ &\Leftrightarrow \exists x\,(x \in P \wedge A(x))\end{aligned}$$

- Inclusion and extensional equality:

$$\begin{aligned}P \subseteq Q &\equiv \forall x\,(x \in P \Rightarrow x \in Q)\\ P = Q &\equiv \forall x\,(x \in P \Leftrightarrow x \in Q)\end{aligned}$$

- Set constructors: $P \cup Q \equiv \{x : x \in P \vee x \in Q\}$ (etc.)

Natural deduction for classical 2nd-order logic (NK2)

Rules of system NK2

$$\frac{}{\Gamma \vdash A}\ {\scriptstyle A\in\Gamma} \qquad \frac{}{\Gamma \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\, A}\ {\scriptstyle x\notin FV(\Gamma)} \qquad \frac{\Gamma \vdash \forall x\, A}{\Gamma \vdash A\{x := e\}}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall X\, A}\ {\scriptstyle X\notin FV(\Gamma)} \qquad \frac{\Gamma \vdash \forall X\, A}{\Gamma \vdash A\{X := P\}}$$

- From these rules, one can derive the introduction & elimination rules for $\bot$, $\wedge$, $\vee$, $\exists^1$, $\exists^2$, $=$ using their 2nd-order definition
- Classical logic obtained via Peirce's law: $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
- Elimination rule for 2nd-order $\forall$ implies all comprehension axioms:

$$\forall \vec{z}\ \forall \vec{Z}\ \exists X\ \forall \vec{x}\ [X(\vec{x}) \ \Leftrightarrow\ A(\vec{x}, \vec{z}, \vec{Z})]$$

A type system for classical 2nd-order logic (λNK2)

- Represent the computational contents of classical proofs using Curry-style proof terms, with call/cc for classical logic:

$$t, u \quad ::= \quad x \quad | \quad \lambda x \,.\, t \quad | \quad tu \quad | \quad \text{cc}$$

- **Typing judgement:** $\underbrace{x_1 : A_1, \ldots, x_n : A_n}_{\text{typing context } \Gamma} \vdash t : B$

Typing rules

$$\frac{}{\Gamma \vdash x : A}\ {\scriptstyle (x:A)\in\Gamma} \qquad \frac{}{\Gamma \vdash \text{cc} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x \,.\, t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall x\, A}\ {\scriptstyle x \notin FV(\Gamma)} \qquad \frac{\Gamma \vdash t : \forall x\, A}{\Gamma \vdash t : A\{x := e\}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X\, A}\ {\scriptstyle X \notin FV(\Gamma)} \qquad \frac{\Gamma \vdash t : \forall X\, A}{\Gamma \vdash t : A\{X := P\}}$$

Note: $\forall$ interpreted uniformly; type checking/inference undecidable

From the derivation to the proof term

- Deduction system NK2 and type system λNK2 are equivalent:

  $A_1, \ldots, A_n \vdash_{\mathrm{NK2}} A \quad \text{iff} \quad x_1 : A_1, \ldots, x_n : A_n \vdash_{\mathrm{NK2}} t : A \quad \text{for some } t$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{[\forall x\,(B(x) \Rightarrow C(x))]}^{\,g}}{B(x) \Rightarrow C(x)} \qquad \dfrac{\dfrac{\overline{[\forall x\,(A(x) \Rightarrow B(x))]}^{\,f}}{A(x) \Rightarrow B(x)} \qquad \overline{[A(x)]}^{\,u}}{B(x)}\,@}{C(x)}\,@}{A(x) \Rightarrow C(x)}\,\lambda u}{\forall x\,(A(x) \Rightarrow C(x))}}{\forall x\,(B(x) \Rightarrow C(x)) \;\Rightarrow\; \forall x\,(A(x) \Rightarrow C(x))}\,\lambda g}{\forall x\,(A(x) \Rightarrow B(x)) \;\Rightarrow\; \forall x\,(B(x) \Rightarrow C(x)) \;\Rightarrow\; \forall x\,(A(x) \Rightarrow C(x))}\,\lambda f
$$

$$\lambda f\,.\,\lambda g\,.\,\lambda u\,.\,g\,(f\,u)$$

Typing examples

- Intuitionistic principles:

$$\begin{array}{rcll}
\textbf{pair} & \equiv & \lambda xyz\,.\,z\,x\,y & : \quad \forall X\,\forall Y\,(X \Rightarrow Y \Rightarrow X \wedge Y) \\
\textbf{fst} & \equiv & \lambda z\,.\,z\,(\lambda xy\,.\,x) & : \quad \forall X\,\forall Y\,(X \wedge Y \Rightarrow X) \\
\textbf{snd} & \equiv & \lambda z\,.\,z\,(\lambda xy\,.\,y) & : \quad \forall X\,\forall Y\,(X \wedge Y \Rightarrow Y) \\
\textbf{refl} & \equiv & \lambda z\,.\,z & : \quad \forall x\,(x = x) \\
\textbf{trans} & \equiv & \lambda xyz\,.\,y\,(x\,z) & : \quad \forall x\,\forall y\,\forall z\,(x = y \Rightarrow y = z \Rightarrow x = z)
\end{array}$$

- Excluded middle, double negation elimination:

$$\begin{array}{rcll}
\textbf{left} & \equiv & \lambda xuv\,.\,u\,x & : \quad \forall X\,\forall Y\,(X \Rightarrow X \vee Y) \\
\textbf{right} & \equiv & \lambda yuv\,.\,v\,y & : \quad \forall X\,\forall Y\,(Y \Rightarrow X \vee Y) \\
\textbf{EM} & \equiv & \text{cc}\,(\lambda k\,.\,\textbf{right}\,(\lambda x\,.\,k\,(\textbf{left}\,x))) & : \quad \forall X\,(X \vee \neg X) \\
\textbf{DNE} & \equiv & \lambda z\,.\,\text{cc}\,(\lambda k\,.\,z\,k) & : \quad \forall X\,(\neg\neg X \Rightarrow X)
\end{array}$$

- De Morgan laws:

$$\begin{array}{rl}
\lambda zy\,.\,z\,(\lambda x\,.\,yx) & : \quad \exists x\,A(x) \;\Rightarrow\; \neg\forall x\,\neg A(x) \\
\lambda zy\,.\,\text{cc}\,(\lambda k\,.\,z\,(\lambda x\,.\,k\,(y\,x))) & : \quad \neg\forall x\,\neg A(x) \;\Rightarrow\; \exists x\,A(x)
\end{array}$$

Axioms of classical 2nd-order arithmetic (PA2)

- Defining equations of all primitive recursive functions:

$$\forall x\,(x+0=x) \qquad \forall x\,(x\times 0=0)$$
$$\forall x\,\forall y\,(x+s(y)=s(x+y)) \qquad \forall x\,\forall y\,(x\times s(y)=x\times y+x)$$
$$\forall x\,(\mathrm{pred}(0)=0) \qquad \forall x\,(x-0=0)$$
$$\forall x\,(\mathrm{pred}(s(x))=x) \qquad \forall x\,\forall y\,(x-s(y))=\mathrm{pred}(x-y)$$

etc.

- Peano axioms:

(P3) $\forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y)$

(P4) $\forall x\,\neg(s(x)=0)$

(P5) $\forall x\,(x\in\mathbb{N})$

- **Remark:** Induction is now a single axiom: (thanks to 2nd-order $\forall$)

$$\begin{aligned}\mathrm{Ind} &\equiv \forall x\,(x\in\mathbb{N})\\ &\Leftrightarrow \forall Z\,[0\in Z\Rightarrow\forall y\,(y\in Z\Rightarrow s(y)\in Z)\Rightarrow\forall x\,(x\in Z)]\end{aligned}$$

The problem of induction

- **Problem:** Induction axiom Ind $\equiv \forall x\,(x \in \mathbb{N})$ is not realizable!
  (Due to uniform interpretation of $\forall$)

- **Solution:** Restrict to $\mathrm{PA2}^- := \mathrm{PA2} - \mathrm{Ind}$ and relativize all 1st-order quantifications to $\mathbb{N}$:

Non-relativized		**Relativized**
$\forall x\, A(x)$	$\rightsquigarrow$	$(\forall x \in \mathbb{N})\, A(x)$ $\forall x\,(x{\in}\mathbb{N}{\Rightarrow}A(x))$
$\exists x\, A(x)$ $\forall Z\,(\forall x\,(A(x){\Rightarrow}Z){\Rightarrow}Z)$	$\rightsquigarrow$	$(\exists x \in \mathbb{N})\, A(x)$ $\forall Z\,(\forall x\,(x{\in}\mathbb{N}{\Rightarrow}A(x){\Rightarrow}Z){\Rightarrow}Z)$

Theorem

If $\mathrm{PA2} \vdash A$, then $\mathrm{PA2}^- \vdash A^{\mathbb{N}}$ ($A^{\mathbb{N}} = A$ relativized to $\mathbb{N}$)

Requires to check that $\mathrm{PA2}^- \vdash (\forall x_1, \ldots, x_k \in \mathbb{N})\,(f(x_1, \ldots, x_k) \in \mathbb{N})$
for all primitive recursive function symbols f

The full standard model of PA2

- **Full standard model of PA2** = Tarski model $\mathscr{M}$ in which:
  - 1st-order variables x are interpreted by natural numbers $n \in \mathbb{N}$
  - 2nd-order variables X are interpreted by all relations $R \subseteq \mathfrak{P}(\mathbb{N}^k)$

($\Rightarrow$, $\forall$ are given the usual Tarski interpretation)

Theorem (Soundness)

If $\quad \mathrm{PA2} \vdash A, \quad$ then $\quad \mathscr{M} \models A$

- More generally, we say that a Tarski model $\mathscr{M}$ of PA2 is:
  - Standard when $\quad \mathbb{N}^{\mathscr{M}} = \mathbb{N}$
    In general, we only have $\quad \mathbb{N}^{\mathscr{M}} \supset \mathbb{N}$ (non standard elements)
  - Full when $\quad (\mathrm{Rel}^k\mathbb{N})^{\mathscr{M}} = \mathfrak{P}((\mathbb{N}^{\mathscr{M}})^k)$
    In general, we only have $\quad (\mathrm{Rel}^k\mathbb{N})^{\mathscr{M}} \subset \mathfrak{P}((\mathbb{N}^{\mathscr{M}})^k)$ (may be countable)
- The full standard model of PA2 is unique, up to unique isomorphism (in the sense of models), but it is uncountable

Plan

Terms, stacks and processes

- Syntax of the language parameterized by
  - A countable set $\mathcal{K} = \{\text{cc}; \ldots\}$ of instructions, containing at least the instruction cc (call/cc)
  - A countable set Π_0 of stack constants (or stack bottoms)

Terms, stacks and processes

Terms	t, u	$::=$	$x \mid \lambda x\,.\,t \mid tu \mid \kappa \mid \mathrm{k}_\pi$	$(\kappa \in \mathcal{K})$
Stacks	π, π'	$::=$	$\alpha \mid t \cdot \pi$	$(\alpha \in \Pi_0,\ t$ closed)
Processes	p, q	$::=$	$t \star \pi$	(t closed)

- A λ-calculus with two kinds of constants:
  - Instructions $\kappa \in \mathcal{K}$, including cc
  - Continuation constants k_π, one for every stack π (generated by cc)
- **Notation:** Λ, Π, $\Lambda \star \Pi$ (sets of closed terms / stacks / processes)

Proof-like terms

- **Proof-like term** $\equiv$ Term containing no continuation constant

Proof-like terms $\quad t, u \quad ::= \quad x \quad | \quad \lambda x\,.\,t \quad | \quad tu \quad | \quad \kappa \qquad (\kappa \in \mathcal{K})$

  - **Idea:** All realizers coming from actual proofs are of this form, continuation constants k_π are treated as paraproofs
  - **Notation:** PL $\equiv$ set of closed proof-like terms

- Natural numbers encoded as proof-like terms by:

Krivine numerals $\quad \overline{n} \equiv \overline{s}^n\,\overline{0} \in \mathrm{PL} \qquad (n \in \mathbb{N})$

writing $\overline{0} \equiv \lambda xy\,.\,x$ and $\overline{s} \equiv \lambda nxy\,.\,y\,(n\,x\,y)$

  - **Note:** Krivine numerals $\not\equiv$ Church numerals, but β-equivalent

The Krivine Abstract Machine (KAM) (1/2)

- We assume that the set $\Lambda \star \Pi$ comes with a preorder $p \succ p'$ of evaluation satisfying the following rules:

Krivine Abstract Machine (KAM)

Push	tu	$\star$	π	$\succ$	t	$\star$	$u \cdot \pi$
Grab	$\lambda x . t$	$\star$	$u \cdot \pi$	$\succ$	$t\{x := u\}$	$\star$	π
Save	$\mathbf{c\!c}$	$\star$	$u \cdot \pi$	$\succ$	u	$\star$	$\mathrm{k}_\pi \cdot \pi$
Restore	k_π	$\star$	$u \cdot \pi'$	$\succ$	u	$\star$	π
		$\ldots$				$\ldots$	

(+ reflexivity & transitivity)

- Evaluation not defined but axiomatized. The preorder $p \succ p'$ is another parameter of the calculus, just like the sets $\mathcal{K}$ and Π_0

- Extensible machinery: can add extra instructions and rules (We shall see examples later)

The Krivine Abstract Machine (KAM) (2/2)

- Rules **Push** and **Grab** implement weak head β-reduction:

Push	$tu \star \pi$	$\succ$	$t \star u \cdot \pi$
Grab	$\lambda x \,.\, t \star u \cdot \pi$	$\succ$	$t\{x := u\} \star \pi$

  - Example: $$(\lambda xy\,.\,t)\,u\,v \star \pi \;\succ\; \lambda xy\,.\,t \star u \cdot v \cdot \pi \;\succ\; t\{x := u\}\{y := v\} \star \pi$$

- Rules **Save** and **Restore** implement backtracking:

Save	$\text{c\!c} \star u \cdot \pi$	$\succ$	$u \star \mathrm{k}_\pi \cdot \pi$
Restore	$\mathrm{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

  - Instruction $\text{c\!c}$ most often used in the pattern

$$\begin{aligned} \text{c\!c}\,(\lambda k\,.\,t) \star \pi &\succ \text{c\!c} \star (\lambda k\,.\,t) \cdot \pi \\ &\succ (\lambda k\,.\,t) \star \mathrm{k}_\pi \cdot \pi \\ &\succ t\{k := \mathrm{k}_\pi\} \star \pi \end{aligned}$$

Representing functions

Definition (function representation)

A partial function $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ is represented by a λ_c-term $\widehat{f} \in \Lambda$ if

$$\widehat{f} \star \bar{n}_1 \cdots \bar{n}_k \cdot u \cdot \pi \quad \succ \quad u \star \overline{f(n_1, \ldots, n_k)} \cdot \pi$$

for all $(n_1, \ldots, n_k) \in \mathrm{dom}(f)$ and for all $u \in \Lambda$, $\pi \in \Pi$

- Call by value encoding:
  - Consumes k values and returns 1 value on the stack
  - Control is given to the extra argument u (continuation, return block)
- Examples:

$$\begin{array}{rcl} \widehat{s} & := & \lambda xk \,.\, k\,(\bar{s}\,x) \\ \widehat{+} & := & \lambda xyk \,.\, y\,k\,(\lambda k'z \,.\, \widehat{s}\,z\,k)\,x \\ \widehat{\times} & := & \lambda xyk \,.\, y\,k\,(\lambda k'z \,.\, \widehat{+}\,z\,x\,k)\,\bar{0} \end{array}$$

Theorem (Representation of recursive functions)

All partial recursive functions are represented in the λ_c-calculus

Example of extra instructions (1/2)

- Numbering terms (or stacks): the instruction quote:

$$\mathrm{quote} \star t \cdot u \cdot \pi \quad \succ \quad u \star \overline{\ulcorner t \urcorner} \cdot \pi$$

where $t \mapsto \ulcorner t \urcorner$ is a fixed bijection from Λ to $\mathbb{N}$

  - Useful to realize the axiom of dependent choices (DC) [Krivine 03]

- Testing syntactic equality: the instruction eq:

$$\mathrm{eq} \star t_1 \cdot t_2 \cdot u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi & \text{if } t_1 \equiv t_2 \\ v \star \pi & \text{if } t_1 \not\equiv t_2 \end{cases}$$

  - Can be implemented using quote

- Non-deterministic choice operator: the instruction fork:

$$\mathrm{fork} \star u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi \\ v \star \pi \end{cases}$$

  - Useful for pedagogy – bad for realizability (collapses to forcing)

Example of extra instructions (2/2)

- The instruction stop:

$$\mathsf{stop} \star \pi \quad \not\succ$$

Stops execution. Final result returned on the stack π

- The instruction print:

$$\mathsf{print} \star \overline{n} \cdot u \cdot \pi \quad \succ \quad u \star \pi \qquad \text{(formal specification)}$$

and prints integer n on standard output (informal specification)

  - Useful to display intermediate results without stopping the machine (Poor man's side effect)

- The instruction hace_mate:

$$\mathsf{hace\_mate} \star u \cdot \pi \quad \succ \quad u \star \pi \quad + \quad \text{hace el mate}$$

Plan

Classical realizability: principles

- **Intuitions:**
  - term = "proof" / stack = "counter-proof"
  - process = "contradiction" (slogan: never trust a classical realizer!)

- Classical realizability model parameterized by a pole $\perp\!\!\!\perp$
  = set of processes closed under anti-evaluation

- Each formula A is interpreted as two sets:
  - A set of stacks $\|A\|$ (falsity value)
  - A set of terms $|A|$ (truth value)

- Falsity value $\|A\|$ defined by induction on A (negative interpretation)

- Truth value $|A|$ defined by orthogonality:

$$|A| \quad = \quad \|A\|^{\perp\!\!\!\perp} \quad = \quad \{t \in \Lambda \ : \ \forall \pi \in \|A\| \quad t \star \pi \in \perp\!\!\!\perp\}$$

Architecture of the realizability model

- The realizability model $\mathscr{M}_{\perp\!\!\!\perp}$ is defined from:
  - The full standard model $\mathscr{M}$ of PA2: the ground model
    (but we could take any model $\mathscr{M}$ of PA2 as well)
  - An instance $(\mathcal{K}, \Pi_0, \succ)$ of the λ_c-calculus
  - A saturated set of processes $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ (the pole)
- Architecture:
  - First-order terms/variables interpreted as natural numbers $n \in \mathbb{N}$
  - Formulas interpreted as falsity values $S \in \mathfrak{P}(\Pi)$
  - k-ary second-order variables (and k-ary predicates) interpreted as falsity functions $F : \mathbb{N}^k \to \mathfrak{P}(\Pi)$.

Formulas with parameters $\quad A, B \quad ::= \quad \cdots \quad | \quad \dot{F}(e_1, \ldots, e_k)$

Add a predicate constant $\dot{F}$ for every falsity function $F : \mathbb{N}^k \to \mathfrak{P}(\Pi)$

Interpreting closed formulas with parameters

Let A be a closed formula (with parameters)

- Falsity value $\|A\|$ defined by induction on A:

$$\|\dot{F}(e_1,\ldots,e_k)\| = F(e_1^{\mathbb{N}},\ldots,e_k^{\mathbb{N}})$$

$$\|A \Rightarrow B\| = |A|\cdot\|B\| = \{t\cdot\pi \;:\; t\in|A|,\;\; \pi\in\|B\|\}$$

$$\|\forall x\ A\| = \bigcup_{n\in\mathbb{N}} \|A\{x:=n\}\|$$

$$\|\forall X\ A\| = \bigcup_{F:\mathbb{N}^n\to\mathfrak{P}(\Pi)} \|A\{X:=\dot{F}\}\|$$

- Truth value $|A|$ defined by orthogonality:

$$|A| = \|A\|^{\perp\!\!\!\perp} = \{t\in\Lambda \;:\; \forall\pi\in\|A\| \quad t\star\pi\in\perp\!\!\!\perp\}$$

The realizability relation

Falsity value $\|A\|$ and truth value $|A|$ depend on the pole $\perp\!\!\!\perp$

$\leadsto$ write them (sometimes) $\|A\|_{\perp\!\!\!\perp}$ and $|A|_{\perp\!\!\!\perp}$ to recall the dependency

Realizability relations

$$t \Vdash A \quad \equiv \quad t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Realizability w.r.t. } \perp\!\!\!\perp\text{)}$$

$$t \Vvdash A \quad \equiv \quad \forall \perp\!\!\!\perp \;\; t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Universal realizability)}$$

From computation to realizability (1/2)

Fundamental idea: The computational behavior of a term determines the formula(s) it realizes:

Example 1: A closed term t is identity-like if:

$$t \star u \cdot \pi \quad \succ \quad u \star \pi \qquad\qquad \text{for all } u \in \Lambda,\ \pi \in \Pi$$

Proposition

If t is identity-like, then $t \Vdash \forall X\,(X \Rightarrow X)$

Proof: Exercise! (Remark: converse implication holds – exercise!)

- Examples of identity-like terms:
  - $\lambda x\,.\,x$, $(\lambda x\,.\,x)\,(\lambda x\,.\,x)$, etc.
  - $\lambda x\,.\,\text{œ}\,(\lambda k\,.\,x)$, $\lambda x\,.\,\text{œ}\,(\lambda k\,.\,k\,x)$, $\lambda x\,.\,\text{œ}\,(\lambda k\,.\,k\,x\,\omega)$, etc.
  - $\lambda x\,.\,\mathrm{quote}\,x\,\lambda n\,.\,\mathrm{unquote}\,n\,(\lambda z\,.\,z)$

From computation to realizability (2/2)

Example 2: Control operators:

$$\begin{array}{rcl} \text{cc} \star t \cdot \pi & \succ & t \star k_\pi \cdot \pi \\ k_\pi \star t \cdot \pi' & \succ & t \star \pi \end{array}$$

- "Typing" k_π: $\quad k_\pi \star t \cdot \pi' \succ t \star \pi$

Lemma

If $\pi \in \|A\|$, then $\mathrm{k}_\pi \Vdash A \Rightarrow B$ (B any)

Proof: Exercise

- "Typing" cc: $\quad \text{cc} \star t \cdot \pi \succ t \star k_\pi \cdot \pi$

Proposition (Realizing Peirce's law)

$\text{cc} \Vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$

Proof: Exercise

Anatomy of the model (1/2)

- **Denotation of universal quantification:**

Falsity value: $$\|\forall x\, A\| \;=\; \bigcup_{n\in\mathbb{N}} \|A\{x := n\}\|$$ (by definition)

Truth value: $$|\forall x\, A| \;=\; \bigcap_{n\in\mathbb{N}} |A\{x := n\}|$$ (by orthogonality)

(and similarly for 2nd-order universal quantification)

- **Denotation of implication:**

Falsity value: $$\|A \Rightarrow B\| \;=\; |A| \cdot \|B\|$$ (by definition)

Truth value: $$|A \Rightarrow B| \;\subseteq\; |A| \to |B|$$ (by orthogonality)

writing $|A| \to |B| \;=\; \{t \in \Lambda \;:\; \forall u \in |A| \;\; tu \in |B|\}$ (realizability arrow)

Anatomy of the model (2/2)

- **Degenerate case:** $\perp\!\!\!\perp = \varnothing$
  - Classical realizability mimics the Tarski interpretation:

Degenerated interpretation

In the case where $\perp\!\!\!\perp = 0$, for every closed formula A:

$$|A| = \begin{cases} \Lambda & \text{if } \mathscr{M} \models A \\ \varnothing & \text{if } \mathscr{M} \not\models A \end{cases}$$

- **Non degenerate cases:** $\perp\!\!\!\perp \neq \varnothing$
  - Every truth value $|A|$ is inhabited:

    If $t_0 \star \pi_0 \in \perp\!\!\!\perp$, then $\mathsf{k}_{\pi_0} t_0 \in |A|$ for all A (paraproof)
  - We shall only consider realizers that are proof-like terms ($\in$ PL)

Plan

Adequacy (1/2)

Aim: Prove the theorem of adequacy

$t : A$ (in the sense of λNK2) implies $t \Vdash A$ (in the sense of realizability)

- Closing typing judgments $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$
  - We close logical objects (1st-order terms, formulas, predicates) using semantic objects (natural numbers, falsity values, falsity functions)
  - We close proof-terms using realizers

Definition (Valuations)

1. A valuation is a function ρ such that
   - $\rho(x) \in \mathbb{N}$ for each 1st-order variable x
   - $\rho(X) : \mathbb{N}^k \to \mathfrak{P}(\Pi)$ for each 2nd-order variable X of arity k
2. Closure of A with ρ written $A[\rho]$ (formula with parameters)

Adequacy

Definition (Adequate judgment, adequate rule)

Given a fixed pole $\perp\!\!\!\perp$:

1. A judgment $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$ is adequate if for every valuation ρ and for all $u_1 \Vdash A_1[\rho], \ldots, u_n \Vdash A_n[\rho]$ we have:

$$t\{x_1 := u_1, \ldots, x_n := u_n\} \Vdash A[\rho]$$

2. A typing rule is adequate if it preserves the property of adequacy (from the premises to the conclusion of the rule)

Theorem

1. All typing rules of λNK2 are adequate
2. All derivable judgments of λNK2 are adequate

Corollary: If $\vdash t : A$ (A closed formula), then $t \Vvdash A$

Extending adequacy to subtyping

Definition (Adequate subtyping judgment)

Judgment $A \leq B$ adequate $\quad \equiv \quad \|B[\rho]\| \subseteq \|A[\rho]\|$ (for all valuations)

Remark: Implies $|A[\rho]| \subseteq |B[\rho]|$ (for all ρ), but strictly stronger

- Some adequate typing/subtyping rules:

$$\frac{}{A \leq A} \qquad \frac{A \leq B \qquad B \leq C}{A \leq C} \qquad \frac{\Gamma \vdash t : A \qquad A \leq B}{\Gamma \vdash t : B}$$

$$\frac{}{\forall x\, A \ \leq\ A\{x := e\}} \qquad \frac{}{\forall X\, A \ \leq\ A\{X := P\}}$$

$$\frac{A \leq B}{A \ \leq\ \forall x\, B}\ x \notin FV(A) \qquad \frac{A \leq B}{A \ \leq\ \forall X\, B}\ X \notin FV(A) \qquad \frac{A' \leq A \qquad B \leq B'}{A \Rightarrow B \ \leq\ A' \Rightarrow B'}$$

$$\frac{}{\forall x\, (A \Rightarrow B) \ \leq\ A \Rightarrow \forall x\, B}\ x \notin FV(A) \qquad \frac{}{\forall X\, (A \Rightarrow B) \ \leq\ A \Rightarrow \forall X\, B}\ X \notin FV(A)$$

- Example: $\underbrace{\forall X\, \forall Y\, (((X \Rightarrow Y) \Rightarrow X) \Rightarrow X)}_{\text{Peirce's law}} \ \leq\ \underbrace{\forall X\, (\neg\neg X \Rightarrow X)}_{\text{DNE}}$

Realizing equalities

- Equality between individuals defined by

$$e_1 = e_2 \;\equiv\; \forall Z\,(Z(e_1) \Rightarrow Z(e_2)) \qquad \text{(Leibniz equality)}$$

Denotation of Leibniz equality

Given two closed first-order terms e_1, e_2 (and a pole $\perp\!\!\!\perp$)

$$\|e_1 = e_2\| \quad = \quad \begin{cases} \|\mathbf{1}\| \;=\; \{t \cdot \pi \;:\; (t \star \pi) \in \perp\!\!\!\perp\} & \text{if } [\![e_1]\!] = [\![e_2]\!] \\ \|\top \Rightarrow \bot\| \;=\; \Lambda \cdot \Pi & \text{if } [\![e_1]\!] \neq [\![e_2]\!] \end{cases}$$

writing $\mathbf{1} \equiv \forall Z\,(Z \Rightarrow Z)$ and $\top \equiv \dot{\varnothing}$

- Intuitions:
  - A realizer of a true equality (in the model) behaves as the identity function $\lambda z\,.\,z$
  - A realizer of a false equality (in the model) behaves as a point of backtrack (breakpoint)

Realizing axioms

Corollary 1 (Realizing true equations)

If $\mathscr{M} \models \forall\vec{x}\,(e_1(\vec{x}) = e_2(\vec{x}))$ (truth in the ground model)

then $\mathbf{I} \equiv \lambda z\,.\,z \Vdash \forall\vec{x}\,(e_1(\vec{x}) = e_2(\vec{x}))$ (universal realizability)

Corollary 2

All defining equations of primitive recursive function symbols $(+,\ -,\ \times,\ /,\ \mathrm{mod},\ \uparrow,\ \mathrm{etc.})$ are universally realized by $\mathbf{I} \equiv \lambda z\,.\,z$

Corollary 3 (Realizing Peano axioms 3 and 4)

$$\begin{array}{rcl} \mathbf{I} & \Vdash & \forall x\,\forall y\,(s(x) = s(y) \Rightarrow x = y) \\ \lambda z\,.\,z\,\mathbf{I} & \Vdash & \forall x\,\neg(s(x) = 0) \end{array}$$

Theorem: If $\mathrm{PA2}^- \vdash A$, then $\theta \Vdash A$ for some $\theta \in \mathrm{PL}$

Realizing true Horn formulas

Definition (Horn formulas)

1. A (positive/negative) literal is a formula L of the form

$$L \;\equiv\; e_1 = e_2 \qquad \text{or} \qquad L \;\equiv\; e_1 \neq e_2$$

2. A (positive/negative) Horn formula is a closed formula H of the form

$$H \;\equiv\; \forall\vec{x}\,[L_1 \Rightarrow \cdots \Rightarrow L_p \Rightarrow L_{p+1}] \qquad (p \geq 0)$$

where $L_1, \ldots, L_p$ are positive; L_{p+1} positive or negative

Theorem (Realizing true Horn formulas) [M. 2014]

If $\mathscr{M} \models H$, then:

$$\mathbf{I} \;\equiv\; \lambda z\,.\,z \;\Vdash\; H \qquad \text{(if } H \text{ positive)}$$

$$\lambda z_1 \cdots z_{p+1}\,.\,z_1\,(\cdots(z_{p+1}\,\mathbf{I})\cdots) \;\Vdash\; H \qquad \text{(if } H \text{ negative)}$$

- All axioms of $\mathrm{PA2}^- := \mathrm{PA2} - \mathrm{Ind}$ are Horn formulas
- Quantifications not relativized to $\mathbb{N}$ $\rightsquigarrow$ H holds for all individuals

Provability, universal realizability and truth

- From what precedes:
  1. A provable $\Rightarrow$ A universally realized (by a proof-like term)
  2. A universally realized $\Rightarrow$ A true (in the full standard model)

$\rightsquigarrow$ Universal realizability: an intermediate notion between provability and truth

- **Beware!**

$$\begin{array}{ccc} \text{Intuitionistic proofs of } A & \subseteq & \text{Classical proofs of } A \\ \cap & & \cap \\ \text{Intuitionistic realizers of } A & \begin{array}{c}\not\subseteq \\ \not\supseteq\end{array} & \text{Classical realizers of } A \end{array}$$

Program extraction

Extracting a program from a proof in PA2

If PA2 $\vdash A$, then there is $\theta \in \mathrm{PL}$ such that $\theta \Vdash A^{\mathbb{N}}$
($A^{\mathbb{N}}$ obtained from A by relativizing all 1st-order quantifications to $\mathbb{N}$)

- **In practice:**
  - Only apply the adequacy theorem to the computationally relevant parts of the proof
  - For the computationally irrelevant parts (i.e. Horn formulas), use 'default realizers' $\rightsquigarrow$ realizer optimization
- **Example 1:** $\lambda xy\,.\,\mathbf{I} \;\Vdash\; (\forall x, y \in \mathbb{N})\,(x + y = y + x)$
- **Example 2:** Fermat's last theorem[1]

$$(\forall x, y, z, n \in \mathbb{N})\,(x \geq 1 \Rightarrow y \geq 1 \Rightarrow n \geq 3 \Rightarrow x^n + y^n \neq z^n)$$

1. realized by: $\lambda xyznu_1u_2u_3v\,.\,u_1\,(u_2\,(u_3\,(v\,\mathbf{I})))$

Plan

Some problems of classical realizability

1. **The specification problem**

   Given a formula A, characterize its universal realizers from their computational behavior

   Specifying Peirce's law [Guillermo-M. 2014]

2. **Witness extraction from classical realizers** (cf next slides)

3. **Realizability algebras + Cohen forcing**

   Realizability algebras: a program to well-order $\mathbb{R}$ [Krivine 2011]

   Forcing as a program transformation [M. 2011]

4. **Models induced by classical realizability**

   What are the interesting formulas that are realized in $\mathscr{M}_{\perp\!\!\!\perp}$ that are not already true in the ground model $\mathscr{M}$?

   Realizability algebras II: new models of ZF + DC [Krivine 2012]

The problem of witness extraction

- **Problem:** Extract a witness from a universal realizer (or a proof)

$$t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$$

i.e. some $n \in \mathbb{N}$ such that $A(n)$ is true

- This is not always possible!

$$t_0 \Vdash (\exists x \in \mathbb{N})\, ((x = 1 \land C) \lor (x = 0 \land \neg C))$$

(C = Continuum hypothesis, Goldbach's conjecture, etc.)

- Two possible compromises:
  - Intuitionistic logic: restrict the shape of the realizer t_0 (by only keeping intuitionistic reasoning principles)
  - Classical logic: restrict the shape of the formula $A(x)$ (typically: Δ^0_0-formulas)

Storage operators (1/2)

- The call-by-value implication:

Formulas $\quad A, B \quad ::= \quad \cdots \quad | \quad \{e\} \Rightarrow A$

with the semantics: $\quad \|\{e\} \Rightarrow A\| \; = \; \{\bar{n} \cdot \pi \; : \; n = e^{\mathbb{N}}, \; \pi \in \|A\|\}$

- From the definition: $\quad e \in \mathbb{N} \Rightarrow A \; \leq \; \{e\} \Rightarrow A$

so that: $\quad \mathbf{I} \Vdash \forall x \forall Z \,[(x \in \mathbb{N} \Rightarrow Z) \Rightarrow (\{x\} \Rightarrow Z)]$ (direct implication)

Definition (Storage operator)

A storage operator is a closed proof-like term M such that:

$$M \Vdash \forall x \forall Z \,[(\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)] \qquad \text{(converse implication)}$$

Theorem (Existence)

Storage operators exist, e.g.: $\quad M \; := \; \lambda fn \,.\, n\, f\, (\lambda hx \,.\, h\,(\bar{s}\, x))\, \bar{0}$

Storage operators (2/2)

- Intuitively, a storage operator

$$M \;\Vdash\; \forall x \forall Z\, [(\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)]$$

is a proof-like term that is intended to be applied to

  - a function f that only accepts values (i.e. intuitionistic integers)
  - a classical integer $t \Vdash n \in \mathbb{N}$ (n arbitrary)

and that evaluates (or 'smoothes') the classical integer t into a value of the form $\bar{n}$ before passing this value to f

- By subtyping, we also have:

$$M \;\Vdash\; \forall Z\, [\forall x\, (\{x\} \Rightarrow Z(x)) \;\Rightarrow\; (\forall x \in \mathbb{N})\, Z(x)]$$

This means that if a property $Z(x)$ holds for all intuitionistic integers, then it holds for all classical integers too

- **Conclusion:** $e \in \mathbb{N} \Rightarrow A$ and $\{e\} \Rightarrow A$ interchangeable

Computing with storage operators

- Given a k-ary function symbol f, we let:

$$\begin{array}{lcl}\mathrm{Total}(f) & := & (\forall x_1 \in \mathbb{N}) \cdots (\forall x_k \in \mathbb{N})(f(x_1, \ldots, x_k) \in \mathbb{N}) \\ \mathrm{Comput}(f) & := & \forall x_1 \cdots \forall x_k\, \forall Z\, [\{x_1\} \Rightarrow \cdots \Rightarrow \{x_k\} \Rightarrow \\ & & \qquad (\{f(x_1, \ldots, x_k)\} \Rightarrow Z) \Rightarrow Z]\end{array}$$

Theorem (Specification of the formula Comput(f))

For all $t \in \Lambda$, the following assertions are equivalent:

1. $t \Vdash \mathrm{Comput}(f)$
2. t computes f: for all $(n_1, \ldots, n_k) \in \mathbb{N}^k$, $u \in \Lambda$, $\pi \in \Pi$:

$$t \star \overline{n}_1 \cdots \overline{n}_k \cdot u \cdot \pi \ \succ\ u \star \overline{f(n_1, \ldots, n_k)} \cdot \pi$$

- Using a storage operator M, we can build proof-like terms:

$$\begin{array}{lcccc}\xi_k & \Vdash & \mathrm{Total}(f) & \Rightarrow & \mathrm{Comput}(f) \\ \xi'_k & \Vdash & \mathrm{Comput}(f) & \Rightarrow & \mathrm{Total}(f)\end{array}$$

The naive extraction method

- A classical realizer $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$ always evaluates to a pair witness/justification:

Naive extraction

If $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$, then there are $n \in \mathbb{N}$ and $u \in \Lambda$ such that:

$$t_0 \star M(\lambda xy\,.\,\mathrm{stop}\,x\,y) \cdot \pi \quad \succ \quad \mathrm{stop} \star \overline{n} \cdot u \cdot \pi$$

(where $u \Vdash A(n)$ w.r.t. the particular pole $\perp\!\!\!\perp$... needed to prove the property)

- But $n \in \mathbb{N}$ might be a false witness because the justification $u \Vdash A(n)$ is cheating! (u might contain hidden continuations)

- In the case where t_0 comes from an intuitionistic proof, extracted witness $n \in \mathbb{N}$ is always correct

  (Can be proved using Kleene realizability adapted to $\mathrm{PA2}^-$)

Extraction in the Σ^0_1-case

Extraction in the Σ^0_1-case (+ display intermediate results)

If $t_0 \Vdash (\exists x \in \mathbb{N})(f(x) = 0)$, then

$$t_0 \star M(\lambda xy \,.\, \mathrm{print}\, x\, y\, (\mathrm{stop}\, x)) \cdot \pi \quad \succ \quad \mathrm{stop} \star \overline{n} \cdot \pi$$

for some $n \in \mathbb{N}$ such that $f(n) = 0$

- Storage operator M used to evaluate 1st component (x)
- 2nd component (y) used as a breakpoint
  (Relies on the particular structure of equality realizers)
- Holds independently from the instruction set
- Supports any representation of numerals
  (One has to implement the storage operator M accordingly)

Example: the minimum principle

- Given a unary function symbol f, write:

$$\mathrm{Total}(f) \quad := \quad (\forall x \in \mathbb{N})(f(x) \in \mathbb{N}) \qquad \text{(totality predicate)}$$

$$x \leq y \quad := \quad x - y = 0 \qquad \text{(truncated subtraction)}$$

Theorem (Minimum principle – MinP)

$$\mathrm{PA2}^- \ \vdash \ \mathrm{Total}(f) \ \Rightarrow \ (\exists x \in \mathbb{N}) \underbrace{(\forall y \in \mathbb{N})\,(f(x) \leq f(y))}_{\text{undecidable}}$$

Proof. Reductio ad absurdum + course by value induction

- The minimum principle is not intuitionistically provable (oracle)
- We cannot apply the Σ^0_1-extraction technique to the above proof (applied to a totality proof of f), since the conclusion is Σ^0_2

  The body $(\forall y \in \mathbb{N})\,(f(x) \leq f(y))$ of $\exists$-quantification is undecidable

Using the minimum principle to prove a Σ^0_1-formula

- **Idea:** The value x given by the minimum principle can be used to prove a Σ^0_1-formula, so that we can perform program extraction:

Corollary

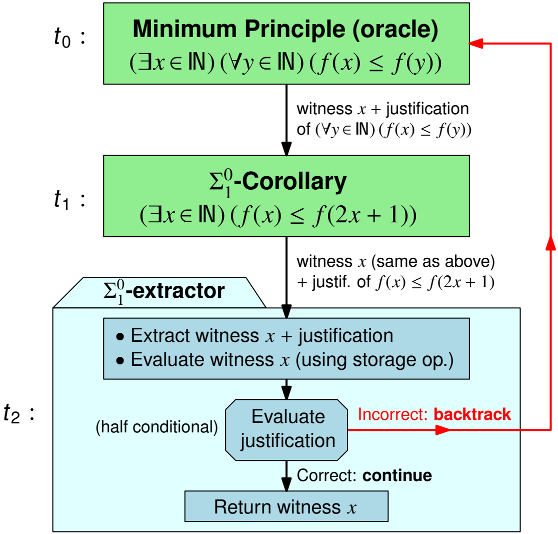
$$\mathrm{PA2}^- \;\vdash\; \mathrm{Total}(f) \;\Rightarrow\; (\exists x \in \mathbb{N})\, \underbrace{(f(x) \leq f(2x+1))}_{\text{decidable}}$$

More generally: $\mathrm{PA2}^- \;\vdash\; \mathrm{Total}(f) \wedge \mathrm{Total}(g) \;\Rightarrow\; (\exists x \in \mathbb{N})\, (f(x) \leq f(g(x)))$

Proof. Take the point x given by the minimum principle

- Applying Σ^0_1-extraction to the above non-constructive proof, we get a correct witness in finitely many evaluation steps
- How is this witness computed?

The algorithm underlying Σ^0_1-extraction



Transcript of the extraction process

Take $f(x) = |x - 1000|$ (real minimum at $x = 1000$)

and apply Σ^0_1-extraction to the proof of $(\exists x \in \mathbb{N})\,(f(x) \leq f(2x+1))$

Step 1 Oracle says: take $x = 0$ since $(\forall y \in \mathbb{N})\,(f(0) \leq f(y))$ (false)
Corollary says: take $x = 0$ since $f(0) \leq f(1)$ (false)
Σ^0_1-extractor evaluates incorrect justification and backtracks

Step 2 Oracle says: take $x = 1$ since $(\forall y \in \mathbb{N})\,(f(1) \leq f(y))$ (false)
Corollary says: take $x = 1$ since $f(1) \leq f(3)$ (false)
Σ^0_1-extractor evaluates incorrect justification and backtracks

Step 3 Oracle says: take $x = 3$ since $(\forall y \in \mathbb{N})\,(f(3) \leq f(y))$ (false)
Corollary says: take $x = 3$ since $f(3) \leq f(7)$ (false)
Σ^0_1-extractor evaluates incorrect justification and backtracks

Step 4 Oracle says: take $x = 7$ since $(\forall y \in \mathbb{N})\,(f(7) \leq f(y))$ (false)

. . . .

Step 11 Oracle says: take $x = 1023$ since $(\forall y \in \mathbb{N})\,(f(1023) \leq f(y))$ (false)
Corollary says: take $x = 1023$ since $f(1023) \leq f(2047)$ (true)
Σ^0_1-extractor evaluates correct justification and returns $x = 1023$

Note that answer $x = 1023$ is correct... but not the point where f reaches its minimum

Extraction in the Σ^0_n-case (1/2)

Definition (Conditional refutation)

$r_A \in \Lambda$ is a conditional refutation of the predicate $A(x)$ if

$$\text{For all } n \in \mathbb{N} \text{ such that } \mathscr{M} \not\models A(n): \quad r_A\,\overline{n} \ \Vdash\ \neg A(n)$$

- Such a conditional refutation can be constructed for every predicate $A(x)$ of 1st-order arithmetic

  This result is a consequence of the following

Theorem (Realizing true arithmetic formulas) [Krivine-Miquey]

For every formula $A(x_1, \ldots, x_k)$ of 1st-order arithmetic, there exists a closed proof-like term t_A such that:

$$\text{If} \quad \mathscr{M} \models A(n_1, \ldots, n_k), \quad \text{then} \quad t_A\,\overline{n}_1 \cdots \overline{n}_k \ \Vdash\ A(n_1, \ldots, n_k)$$

(for all $n_1, \ldots, n_k \in \mathbb{N}$)

Extraction in the Σ^0_n-case (2/2)

The Kamikaze extraction method [M. 2009]

Let

1. $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$
2. r_A a conditional refutation of the predicate $A(x)$

Then the process

$$t_0 \star M\,(\lambda xy\,.\,\mathrm{print}\,x\,(r_A\,x\,y)) \cdot \pi$$

displays a correct witness after finitely many evaluation steps

- **Remark:** No correctness invariant is ensured as soon as the (first) correct witness has been displayed!

  After, anything may happen: crash, infinite loop, displaying incorrect witnesses, etc. (Kamikaze behavior)

Interlude: on numeration systems

- Numeration systems used in the History:

Tally sticks	(35000 BC)	
Babylonian	(3100 BC)	
Egyptian	(3000 BC)	∩∩∩∩II
Roman	(1000 BC)	XLII
Hindu-Arabic	(300 AD)	42

- Numeration systems used in Logic:

Peano: $ssssssssssssssssssssssssssssssssssssssssss0$

Church: $\lambda xf\,.\,f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f($
$f(f(f(f(f(f(f(f(f(f(f(f\,x)))$

Krivine:
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))($
$(\lambda xf.x))$

Primitive numerals (1/2)

To get rid of Krivine numerals $\bar{n} = \bar{s}^n\bar{0}$ (cf paleolithic numeration)
we extend the machine with the following instructions:

- For every natural number $n \in \mathbb{N}$, an instruction $\widehat{n} \in \mathcal{K}$ with no evaluation rule (i.e. inert constant: pure data)

  Intuition: $\widehat{n} \star \pi \succ$ `segmentation fault`

- An instruction null $\in \mathcal{K}$ with the rules

$$\mathrm{null} \star \widehat{n} \cdot u \cdot v \quad \succ \quad \begin{cases} u \star \pi & \text{if } n = 0 \\ v \star \pi & \text{otherwise} \end{cases}$$

- Instructions $\check{f} \in \mathcal{K}$ with the rules

$$\check{f} \star \widehat{n}_1 \cdots \widehat{n}_k \cdot u \cdot \pi \quad \succ \quad u \star \widehat{m} \cdot \pi \qquad \text{where } m = f(n_1, \ldots, n_k)$$

  for all the usual arithmetic operations

Primitive numerals (2/2)

- Call-by-value implication, yet another definition:

Formulas $\quad A, B \quad ::= \quad \cdots \quad | \quad [e] \Rightarrow A$

with the semantics: $\quad \|\{e\} \Rightarrow A\| \; = \; \{\widehat{n} \cdot \pi \;:\; n = e^{\mathbb{N}},\; \pi \in \|A\|\}$

- Redefining the set of natural numbers:

$$\mathbb{N}' \;:=\; \{x \;:\; \forall Z\,(([x] \Rightarrow Z) \Rightarrow Z)\}$$

$$\begin{array}{lll}
\mathsf{box} \;:=\; \lambda k\,.\,k\,x & \Vdash & \forall x\,([x] \Rightarrow x \in \mathbb{N}') \\
\mathsf{box}\,\widehat{n} & \Vdash & n \in \mathbb{N}' \\
\lambda n\,.\,n\,\lambda x\,.\,\check{s}\,x\,\mathsf{box} & \Vdash & (\forall x \in \mathbb{N}')(s(x) \in \mathbb{N}') \\
\lambda nm\,.\,n\,\lambda x\,.\,m\,\lambda y\,.\,(\check{+})\,x\,y\,\mathsf{box} & \Vdash & (\forall x, y \in \mathbb{N}')(x + y \in \mathbb{N}')
\end{array}$$

$$\begin{array}{rl}
\mathsf{rec_cbv} \;:= & \lambda z_0 z_s\,.\,\mathbf{Y}\,\lambda rx\,.\,\mathsf{null}\,x\,z_0\,((\check{-})\,x\,\hat{1}\,\lambda y\,.\,z_s\,y\,(r\,y)) \\
\Vdash & \forall Z\,[Z(0) \;\Rightarrow\; \forall y\,([y] \Rightarrow Z(y) \Rightarrow Z(s(y))) \;\Rightarrow\; \forall x\,([x] \Rightarrow Z(x))] \\
\mathsf{rec} \;:= & \lambda z_0 z_s n\,.\,n\,\lambda x\,.\,\mathsf{rec_cbv}\,z_0\,(\lambda yz\,.\,z_s\,(\mathsf{box}\,y)\,z)\,x \\
\Vdash & \forall Z\,[Z(0) \Rightarrow (\forall y \in \mathbb{N}')(Z(y) \Rightarrow Z(s(y))) \;\Rightarrow\; (\forall x \in \mathbb{N}')Z(x)]
\end{array}$$

- **Conclusion:** $\quad \Vdash\; \forall x\,(x \in \mathbb{N}' \;\Leftrightarrow\; x \in \mathbb{N})$

Krivine's realizability vs the LRS-translation (1/2)

- Krivine's realizability can be seen as the composition of the Lafont-Reus-Streicher (LRS) translation with Kleene realizability:

$$\text{CPS} \circ \text{Krivine} \quad = \quad \text{Kleene} \circ \text{LRS}$$

[Oliva-Streicher 2008]

The dictionary

Classical realizability (Krivine)	Lafont-Reus-Streicher translation
Pole $\perp\!\!\!\perp$	Return formula R
Falsity value $\|A\|$	Negative translation $A^{\perp}$
$\|A \Rightarrow B\| \ := \ \lvert A\rvert \cdot \|B\|$	$(A \Rightarrow B)^{\perp} \ := \ A^{LRS} \wedge B^{\perp}$
Truth value $\lvert A\rvert \ := \ \|A\|^{\perp\!\!\!\perp}$	$A^{LRS} \ := \ A^{\perp} \Rightarrow R$

- Through the CPS-translation, Krivine's extraction method in the Σ^0_1-case is exactly Friedman's trick (transposed to LRS) [M. 2010]

Krivine's realizability vs the LRS-translation (2/2)

Beware of reductionism!

- The decomposition holds only for *pure* classical reasoning
  (extra instructions are not taken into account)
- Classical realizers are easier to understand than their CPS-translations (and more efficient)
- Classical realizability is more than Kleene's realizability composed with the Lafont-Reus-Streicher translation

An image:

$$2H_2 + O_2 \quad \longrightarrow \quad 2H_2O$$

but can we deduce the properties of *water* from the ones of H_2 and O_2?